UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/716,099	11/17/2003	Byron D. Vargas	03917-P0001B	6113

24126 7590 11/02/2006

ST. ONGE STEWARD JOHNSTON & REENS, LLC
986 BEDFORD STREET
STAMFORD, CT 06905-5619

EXAMINER	
NGUYEN, PHILLIP H	

ART UNIT	PAPER NUMBER
2191	

DATE MAILED: 11/02/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/716,099	VARGAS, BYRON D.	
	Examiner	Art Unit	
	Phillip H. Nguyen	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *17 November 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-48* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-48* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on _____ is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *20031117*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the original filing of November 20, 2003. Claims 1-48 are pending and have been considered below.

Claim Rejections - 35 USC § 101

2. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 1, 14 and 43 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. These claims are non-statutory because the language of the claims is directed to software, lacking storage on a medium, which enables any underlying functionality to occur.

Claim Rejections - 35 USC § 112

3. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

4. Claims 3, 6, 16, 35, 44 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claims 3, 16, 35, and 44 recite the limitation "substantially identical" and it is unclear to the examiner how substantially the re-translated first computer language source code identical to the first computer language source code.

Claim 6 recites the limitation "class consists of units selected from the group consisting of..." the first consists of is unclear to the examiner since the second "consisting" means the class can be any one of a method, or data fields or inner classes, or any combination of those, whereas the first "consists of" attempts to limit it to something very specific, but does not specify which one of all those choices.

Claim Rejections - 35 USC § 102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

6. Claims 1-11, 43-45, and 47-48 are rejected under 35 U.S.C. 102(b) as being anticipated by Simser (US 6,314,429 B1).

Claim 1: Simser discloses a computer language translator for translating a first computer language source code to a second computer language source code, comprising:

a. a computer ("conversion system" Col 2, line 41);

b. a library accessible by said computer ("conversion library", Col 2, line 44), said library including data indicative of types of data manipulations between the first computer language source code ("garbage collection is responsible for managing the memory allocated for the string object" Col 3, line 66-67) and the second computer language source code ("free of a string returns memory back to the computer allowing it to be reused later" Col 4, line 1-2); and

c. software executing on said computer ("Java2c and c2Java" Col 6, line 51), said software analyzing the first computer language source code to identify a type of data manipulation that the first computer language source code performs ("identify the required C data structure" Col 6, line 63), accessing said library and correlating the type of data manipulation the first computer language source code performs to second computer language source code ("create a java class with data members in a one-to-one (correlating) correspondence with the C data structure identified in step 1" Col 7, line 1-2), the correlation being independent of the context in which the first computer source code is used ("conversion library is not dependent on the name of functions and the choice of names is also not restricted" Col 3, line56-58, this means that the mapping ("correlating") between C programming language and Java programming language is independent of the context), said software generating second computer language source code that emulates the type of data manipulation the first computer language source code performs ("creating a Java class to parallel (emulates) a C structure and describing the Java class via a class descriptor" Col 6, line 60-62).

Claim 2: Simser discloses the computer language translator as in claim 1 above, and further discloses the translator is a bi-directional translator ("Bi-directional conversion library" Col 6, line 21), all the other limitations are substantially identical to claim 1 above, and, therefore, are rejected under the same reason.

Claim 3: Simser discloses the computer language translator as in claim 2 above, but does not explicitly disclose the re-translated first computer language source code is substantially identical to the first computer language source code. It is inherent in Simser's approach in order to fulfill the purpose of bi-directional conversion.

Claim 4: Simser discloses the computer language translator as in claim 1 above, and further disclose software performs a name adjustment when an incompatibility between the first computer language source code and the second computer language source code occurs during correlation (mapping) of the type of data manipulation the first computer language source code performs to the second computer language source code to resolved the incompatibility ("the conversion library (software) provides a suite of functions to provide a consistent, well-defined approach to implementing the mapping" Col 2, line 49-51), the name adjustment being independent of the context in which the incompatibility is located ("conversion library is not dependent on the name of functions and the choice of names is also not restricted" Col 3, line56-58).

Claim 5: Simser discloses the computer language translator as in claim 1 above, and further discloses the first computer language source code comprises a class ("via a class descriptor" Col 6, line 62).

Claim 6: Simser discloses the computer language translator as in claim 5 above, and further discloses the class consists of units selected from the group consisting of: method, data fields, inner-classes, and combination thereof (Col 7, line 10-65).

Claim 7: Simser discloses the computer language translator as in claim 1 above, and further discloses the first computer language source code comprises an identifier ("allocCString" Col 4, line 36, identifier is any text string used as a label, such as the name of a procedure or a variable in a program).

Claim 8: Simser discloses the computer language translator as in claim 7 above, and further discloses software performs a name adjustment when an incompatibility between the identifier and the second computer language source code occurs during correlation of the type of data manipulation the identifier performs to the second computer language source code to resolve the incompatibility ("since the mapping for each function may be different (e.g. a single Java data object may map into multiple parameters in the function signature). The conversion library provides a suite of functions (identifiers) to provide a consistent, well-defined approach in implementing the mapping layer" Col 2, line 46-51), the name adjustment being independent of the

context in which the identifier is used ("conversion library is not dependent on the name of functions ("identifiers") and the choice of names is also not restricted" Col 3, line56-58).

Claim 9: Simser discloses the computer language translator as in claim 1 above, and further discloses the software generates a tagged element ("JNIEnv * env, Kstromg kavaStr" Col 4, line 37-38) inserted in the second computer language source code indicative of a type of data manipulation the first computer language source code performs ("creates a new C string based on a passed Java string object" Col 4, line 26-27)

Claim 10: Simser discloses the computer language translator as in claim 9 above, and further discloses the tagged element comprises information selected from the group consisting of: formatting, translating data, and first computer language source code (Col 4, line 46-50).

Claim 11: Simser discloses the computer language translator as in claim 1 above, and further discloses the first computer language is Java and the second computer language is C++ (Col 2, line 19)

Claim 43: Simser discloses a computer language translating system for translating a Java source code to an OOP language source code, comprising:

a. a library ("conversion library" Col 2, line 44) including data indicative of types of data manipulations between the Java source code and the OOP language source code ("convert Java data objects into equivalent C structures" Col 2, line 44-45, manipulating java objects is different than manipulating C structures); and

b. although, Simser discloses identify the required C data structure (Col 6, line 63), but does not explicitly disclose an analyzer for analyzing the Java source code to determine a type of data manipulation that the first computer language source code performs. It is inherent in Simser's system since his computer language translating system support bi-directional translating; accessing said library and correlating the type of data manipulation the Java source code performs to OOP language source code ("create a java class with data members in a one-to-one (correlating) correspondence with the C data structure identified in step 1" Col 7, line 1-2); the correlation being independent of the context in which the first computer source code is used ("conversion library is not dependent on the name of functions and the choice of names is also not restricted" Col 3, line56-58, this means that the mapping ("correlating") between OOP language source code and Java source code is independent of the context); and

c. a generator for generating OOP language source code that emulates the type of data manipulation the Java source code performs ("creating a Java class to parallel (emulates) a C structure" Col 6, line 60-62).

Claim 44: Simser discloses the computer language translating system as in claim 43 above, and further discloses analyzer analyzes the OOP language source

code to determine a type of data manipulation the OOP language source code performs ("identify the required C data structure" Col 6, line 63); said analyzer accessing said library to correlate (map) the type of data manipulation the OOP language source code performs to Java source code ("create a java class with data members in a one-to-one (correlating) correspondence with the C data structure identified in step 1" Col 7, line 1-2); the correlation (mapping) being independent of the context in which the OOP language source code is used ("conversion library is not dependent on the name of functions and the choice of names is also not restricted" Col 3, line56-58, this means that the mapping ("correlating") between OOP language source code and Java source code is independent of the context); and said generator generates re-translated Java source code that emulates the type of data manipulation the OOP language source code performs ("creating a Java class to parallel (emulates) a C structure" Col 6, line 60-62); the re-translated Java source code being substantially identical to the Java source code (the re-translated Java source code is substantially identical to the java source otherwise it will defeat the purpose of bi-directional translating).

Claim 45: Simser disclose the computer language translating system as in claim 43 above, and further discloses C and C++ supported by Java Native Interface (Col 2, line 19), but does not explicitly disclose analyzer builds class declarations and class definitions based on the type of data manipulation the Java source code performs and said generator generates OOP language source code based upon the class declarations and class definitions. It is inherent in Simser's translating system in order

to translate between C++ and Java since his system is a bi-directional translating system.

Claim 47: Simser discloses a method for translating a first computer language source code that performs a data manipulation with a virtual machine to a second computer language source code that performs a data manipulation without used of a virtual machine, comprising the steps of:

a. although, Simser does not explicitly disclose analyzing the first computer language source code to determine a type of data manipulation the first computer language source code performs. It is inherent in Simser's approach in order to performs the conversion from Java to C

b. correlating (mapping) the type of data manipulation of the first computer language source code performs to the second computer language source code ("creating a Java class with data members in a one-to-one (correlating) correspondence with the C structure" Col 7, line 1-2);

c. generating the second computer language source code, the second computer language source code emulating the type of data manipulation the first computer language source code performs ("the mapping layer 16 will take any Java data submitted from the calls from the API layer 22 located inside the middle layer and translate the Java data 12 into a equivalent C data structure" Col 3, line 6-9);

d. performing a data manipulation with the second computer language source code that emulates the type of data manipulation the first computer language source

code performs ("free of a string returns memory back to the computer allowing it to be reused later" Col 4, line 1-2), the second computer language source code performing the data manipulation without use of a virtual machine (C programming language does not use virtual machine).

Claim 48: Simser discloses a method for translating a first computer language source code that performs a data manipulation with a garbage collector to a second computer language source code that performs a data manipulation without used of a garbage collector, comprising the steps of:

a. although, Simser does not explicitly disclose analyzing the first computer language source code to determine a type of data manipulation the first computer language source code performs. It is inherent in Simser's approach in order to performs the conversion from Java to C

b. correlating (mapping) the type of data manipulation of the first computer language source code performs to the second computer language source code ("creating a Java class with data members in a one-to-one (correlating) correspondence with the C structure" Col 7, line 1-2);

c. generating the second computer language source code, the second computer language source code emulating the type of data manipulation the first computer language source code performs ("the mapping layer 16 will take any Java data submitted from the calls from the API layer 22 located inside the middle layer and translate the Java data 12 into a equivalent C data structure" Col 3, line 6-9);

d. performing a data manipulation with the second computer language source code that emulates the type of data manipulation the first computer language source code performs ("free of a string returns memory back to the computer allowing it to be reused later" Col 4, line 1-2), the second computer language source code performing the data manipulation without use of a virtual machine ("there is no garbage collection in C, all strings created with allocCString must be freed via a call to freeCString" Col 4, line 41-43).

7. Claims 14, 17-19, 25, 27-29, 34, 36-38, and 46 are rejected under 35 U.S.C. 102(b) as being anticipated by Andrews et al (5,768,564).

Claim 14: Andrews discloses a computer language translating system for translating a first computer language source code to a second computer language source code, comprising:

a. a library ("collection of tokens" Col 8, line 20) accessible by said translating system, said library including data indicative of types of data manipulation between the first computer language source code and the second computer language source code ("encapsulating" Col 7, line 11, encapsulating is one type of data manipulation);

b. a parser for parsing the first computer language source code into parsed elements (Figure 4, item 5);

c. an analyzer for analyzing the parsed elements to identify the type of data manipulation that the first computer language source code performs ("semantic

analyzer" Figure4, item 17) analyzer accessing said library and correlating ("mapping") the type of data manipulation the first computer language source code performs to second computer language source code, the correlation being independent of the context in which the first computer language source code is used ("each source language macro maps to exactly one target language macro, it checks the textual consistency of every macro use to ensure that the macro text works in all contexts of use" Col 6, line 17-20); and

d. a generator for generating second computer language source code that emulates the type of data manipulation the first computer language source code performs ("generate simple resultant code to emulate simple source code" Col 12, line 11).

Claim 17: Andrews discloses the computer language translating system as in claim 14 above, and further discloses generator performs a name adjustment when an incompatibility between the identifier and the second computer language source code occurs during generation of the second computer language source code (Col 11, line 52-55), the name adjustment being independent of the context in which the incompatibility is located ("each source language macro maps to exactly one target language macro, it checks the textual consistency of every macro use to ensure that the macro text works in all contexts of use" Col 6, line 17-20).

Claim 18: Andrews discloses the computer language translating system as in claim 14 above, and further discloses the first computer language source code comprises an identifier ("s, i, j, b, or a" Col 12, line 25-30, variables are identifiers).

Claim 19: Andrews discloses the computer language translating system as in claim 18 above, and further discloses generator performs a name adjustment when an incompatibility between the identifier and the second computer language source code occurs during generation of the second computer language source code (Col 11, line 52-55), the name adjustment being independent of the context in which the incompatibility is located ("each source language macro maps to exactly one target language macro, it checks the textual consistency of every macro use to ensure that the macro text works in all contexts of use" Col 6, line 17-20).

Claim 25: Andrews discloses a method for translating a first computer language source code to a second computer language source code comprising the step of:

a. inputting the first computer language source code into a computer ("source code" Figure 4, item 1, a stream of tokens);

b. parsing the first computer language source code into parsed elements ("parser" Figure 4, item 5, parsing the stream of tokens to create a syntax tree representing a program);

c. analyzing the parsed elements to determine the type of data manipulation the first computer language source code performs ("semantic analyzer" Figure 4, item 17, analyzing the macro);

d. building class declarations and class definitions from the parsed elements (Col 12, line 25-34) that are independent of the context of the first computer language source code("to ensure that the macro text works in all contexts of use", Col 6, line 19-20); and

e. generating the second computer language source code according to the class declarations and class definitions (Col 12, line 25-34), the second computer language source code emulating the type of data manipulation the first computer language source code performs ("generate simple resultant code to emulate simple source code" Col 12, line 10).

Claim 27: Andrews discloses the translation method as in claim 25 above, and further discloses generator performs a name adjustment when an incompatibility between the identifier and the second computer language source code occurs during generation of the second computer language source code (Col 11, line 52-55), the name adjustment being independent of the context in which the incompatibility is located ("each source language macro maps to exactly one target language macro, it checks the textual consistency of every macro use to ensure that the macro text works in all contexts of use" Col 6, line 17-20).

Claim 28: Andrews discloses the computer language translating system as in claim 25 above, and further discloses the first computer language source code comprises an identifier ("s, i, j, b, or a" Col 12, line 25-30, variables are identifiers).

Claim 29: Andrews discloses the translation method as in claim 28 above, and further discloses generator performs a name adjustment when an incompatibility between the identifier and the second computer language source code occurs during generation of the second computer language source code (Col 11, line 52-55), the name adjustment being independent of the context in which the incompatibility is located ("each source language macro maps to exactly one target language macro, it checks the textual consistency of every macro use to ensure that the macro text works in all contexts of use" Col 6, line 17-20).

Claim 34: Andrews disclose a method for translating a first computer language source code to a second computer language source code comprising the steps of:

a. analyzing the first computer language source code to determine a type of data manipulation that the first computer language source code performs ("semantic analyzer" Figure4, item 17);

b. correlating ("mapping") the type of data manipulation the first computer language source code performs to second computer language source code, the correlation being independent of the context in which the first computer language source code is used ("each source language macro maps to exactly one target

language macro, it checks the textual consistency of every macro use to ensure that the macro text works in all contexts of use" Col 6, line 17-20); and

c. generating second computer language source code, the second computer language source code emulating the type of data manipulation the first computer language source code performs ("generate simple resultant code to emulate simple source code" Col 12, line 11).

Claim 36: Andrews discloses the translation method as in claim 34 above, and further discloses generator performs a name adjustment when an incompatibility between the identifier and the second computer language source code occurs during generation of the second computer language source code (Col 11, line 52-55), the name adjustment being independent of the context in which the incompatibility is located ("each source language macro maps to exactly one target language macro, it checks the textual consistency of every macro use to ensure that the macro text works in all contexts of use" Col 6, line 17-20).

Claim 37: Andrews discloses the computer language translating system as in claim 34 above, and further discloses the first computer language source code comprises an identifier ("s, i, j, b, or a" Col 12, line 25-30, variables are identifiers).

Claim 38: Andrews discloses the translation method as in claim 36 above, and further discloses generator performs a name adjustment when an incompatibility

between the identifier and the second computer language source code occurs during generation of the second computer language source code (Col 11, line 52-55), the name adjustment being independent of the context in which the incompatibility is located ("each source language macro maps to exactly one target language macro, it checks the textual consistency of every macro use to ensure that the macro text works in all contexts of use" Col 6, line 17-20).

Claim 46: Andrews discloses a method for translating a portion of Java source code that comprises at least one class to an OOP language source code comprising the steps of:

a. parsing the at least one class into multiple identifiers (Figure 4, item 5);

b. parsing one of the identifiers into name segments (Figure 4, item 5);

c. adjusting the name segments individually and independently from each other according to the OOP language source code ();

d. generating OOP language source code based on the name segments where only a portion of the Java source code is translated to the OOP language source code ("build the C++ fragment tree based on the position in the pTAL fragment tree of the source language token" Col 8, line 38-39); the OOP language source code emulating a type of data manipulation the portion of the Java source code performs ("generate simple resultant code to emulate simple source code" Col 12, line 11).

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

9. Claims 12 and 13 are rejected under 35 U.S.C. 103(a) as being unpatentable over Simser (US 6,314,429).

Claim 12: Simser discloses the computer language translator as in claim 1 above, and further discloses convert data structures from C to Java or with appropriate substitutions of functions, convert between two other different computer programming languages (Col 2, line 55-57), but does not explicitly discloses the first computer language is Java and the second computer language is C#. However, it would have been obvious to one having an ordinary skill in the art at the time the invention was made to recognize that converting between Java and C# or two other different programming languages is supported by Simser's conversion system. Therefore, one having an ordinary skill in the art would have been motivated to use C# as a second computer language in Simser's conversion system because C# is a cocktail of Java.

Claim 13: Simser discloses the computer language translator as in claim 1 above, and further discloses C and C++ supported by Java Native Interface (Col 2, line 19), but does not explicitly disclose the first computer language is C# and the second

computer language is C++. However, it would have been obvious to one having an ordinary skill in the art at the time the invention was made to recognize that converting between two other different programming languages is supported by Simser's conversion system. Therefore, one having an ordinary skill in the art would have been motivated to use C# as a first computer language and C++ is a second computer language in Simser's approach because C# is a cocktail of C++.

10. Claims 15, 16, 20-24, 26, 30-33, 35, and 39-42 rejected under 35 U.S.C. 103(a) as being unpatentable over Andrews (5,768,564) in view of Simser (US 6,314,429).

Claim 15: Andrews discloses the computer language translating system as in claim 14 above, but does not discloses the translating system is a bi-directional translator. However, Simser discloses an analogous translating system performs bi-directional translating. It is obvious to one having an ordinary skill in the art at the time the invention was made to combine Andrews's translating system with Simser's approach. Therefore, the combination is obvious because one having an ordinary skill in the art would have been motivated to include the bi-directional translating feature in Andrews's system if the user decides to view or run the original computer language source code after the translation.

Claim 16: Andrews and Simser disclose the computer language translating system as in claim 15 above, but does not explicitly discloses the re-translated first

computer language source code is substantially identical to the first computer language source code. However, it would have been obvious to one having an ordinary skill in the art at the time the invention was made to recognize that the re-translated first computer language source code is substantially identical to the first computer language source code in order fulfill the purpose of Simser's approach. Therefore, one having an ordinary skill would have been motivated to include this feature in Andrews and Simser's approaches to ensure the bi-directional translating system fulfills its purpose.

Claim 20: Andrews discloses the computer language translating system as in claim 14 above, but does not explicitly disclose the generator generates a tagged element inserted in the second computer language source code indicative of the type of data manipulation the first computer language source code performs. However Simser discloses an analogous bi-directional computer language translating system that generates a tagged element as claimed in the instant claim (Col 4, line 25-50). It would have been obvious to one having an ordinary skill in the art at the time the invention was made to use Simser's approach in Andrews's system. Therefore, one having an ordinary skill would have been motivated to generate tagged element inserted in the second computer language source code for creating the second computer language source code based on the tagged element (Col 4, line 26-28).

Claim 21: Andrews and Simser disclose the computer language translating system as in claim 20 above, and Simser further discloses wherein the tagged element

comprises information selected from the group consisting of: formatting, translation data, and first computer language source code (Col 4, line 48-50).

Claim 22: Andrews discloses the computer language translating system as in claim 14 that translates source code from one high-level computer language to the source code of another language above, but does not explicitly disclose the first computer language is Java. However Simser discloses an analogous translating system that performs bi-directional translating between Java, C, and C++ (Col 2, line 19). It would have been obvious to one having an ordinary skill in the art at the time the invention was made to combine Simser's approach in Andrews's translating system. The combination is obvious because one having an ordinary skill in the art would have been motivated to use Java as the first computer language to translate to C++ because Java Native Interface is supported C++(see Simser Col 2, line 19).

Claim 23: Andrews discloses the computer language translating system as in claim 14 above that translates source code from one high-level computer language to the source code of another language, but does not explicitly disclose the first computer language is Java and the second computer language is C#. However, Simser discloses an analogous translating system that translates bi-direction between Java, C, and C++ (Col 2, line 19). It would have obvious to one having an ordinary skill in the art at the time the invention was made to combine Simser's approach and Andrews's approach. Therefore, one having an ordinary skill in the art would have been motivated to translate

Java programming language to C# programming language since both of them are high-level computer languages and C# is a cocktail of Java.

Claim 24: Andrews discloses the computer language translating system as in claim 14 above that translates source code from one high-level computer language to the source code of another language, but does not explicitly disclose the first computer language is C# and the second computer language is C++. However, Simser discloses an analogous translating system that translates bi-direction between Java, C, and C++. It would have obvious to one having an ordinary skill in the art at the time the invention was made to combine Simser's approach and Andrews's approach. Therefore, one having an ordinary skill in the art would have been motivated to translate C# programming language to C++ programming language since both of them are high-level computer languages and C# is a cocktail of C++.

Claim 26: Andrews discloses the translation method as in claim 25 above; but does not disclose the bi-directional translation as claimed. However, Simser discloses an analogous bi-directional translation method as claimed in the instant clam (Col 2, line 43). It would have been obvious to one having an ordinary skill in the art at the time the invention was made to combine Andrews's method with Simser's bi-directional translation. One having an ordinary skill in the art would have been motivated to combine Simser's method with Andrews's method in order to enable a programmer to go back and forth between two languages without creating a new program.

Claim 30: Andrews and Simser disclose the translation method as in claim 25 above, and Simser further discloses generating a tagged element indicative of the type of data manipulation the first computer language source code performs and inserting the tagged element in the second computer language source code (Col 4, line 35-50).

Claim 31: Andrews discloses the computer language translating system as in claim 25 that translates source code from one high-level computer language to the source code of another language above, but does not explicitly disclose the first computer language is Java. However Simser discloses an analogous translating system that performs bi-directional translating between Java, C, and C++ (Col 2, line 19). It would have been obvious to one having an ordinary skill in the art at the time the invention was made to combine Simser's approach in Andrews's translating system. The combination is obvious because one having an ordinary skill in the art would have been motivated to use Java as the first computer language to translate to C++ because Java Native Interface is supported C++(see Simser Col 2, line 19).

Claim 32: Andrews discloses the computer language translating system as in claim 25 above that translates source code from one high-level computer language to the source code of another language, but does not explicitly disclose the first computer language is Java and the second computer language is C#. However, Simser discloses an analogous translating system that translates bi-direction between Java, C, and C++ (Col 2, line 19). It would have obvious to one having an ordinary skill in the art at the

time the invention was made to combine Simser's approach and Andrews's approach. Therefore, one having an ordinary skill in the art would have been motivated to translate Java programming language to C# programming language since both of them are high-level computer languages and C# is a cocktail of Java.

Claim 33: Andrews discloses the computer language translating system as in claim 25 above that translates source code from one high-level computer language to the source code of another language, but does not explicitly disclose the first computer language is C# and the second computer language is C++. However, Simser discloses an analogous translating system that translates bi-direction between Java, C, and C++. It would have obvious to one having an ordinary skill in the art at the time the invention was made to combine Simser's approach and Andrews's approach. Therefore, one having an ordinary skill in the art would have been motivated to translate C# programming language to C++ programming language since both of them are high-level computer languages and C# is a cocktail of C++.

Claim 35: Andrews discloses the translation method as in claim 34 above, but does not disclose the bi-directional translation as claimed. However, Simser discloses an analogous bi-directional translation method as claimed in the instant clam (Col 2, line 43). It would have been obvious to one having an ordinary skill in the art at the time the invention was made to combine Andrews's method with Simser's bi-directional translation. One having an ordinary skill in the art would have been motivated to

combine Simser's method with Andrews's method in order to enable a programmer to go back and forth between two languages without creating a new program.

Claim 39: Andrews and Simser disclose the translation method as in claim 34 above, and Simser further discloses generating a tagged element indicative of the type of data manipulation the first computer language source code performs and inserting the tagged element in the second computer language source code (Col 4, line 35-50).

Claim 40: Andrews discloses the computer language translating system as in claim 34 that translates source code from one high-level computer language to the source code of another language above, but does not explicitly disclose the first computer language is Java. However Simser discloses an analogous translating system that performs bi-directional translating between Java, C, and C++ (Col 2, line 19). It would have been obvious to one having an ordinary skill in the art at the time the invention was made to combine Simser's approach in Andrews's translating system. The combination is obvious because one having an ordinary skill in the art would have been motivated to use Java as the first computer language to translate to C++ because Java Native Interface is supported C++(see Simser Col 2, line 19).

Claim 41: Andrews discloses the computer language translating system as in claim 34 above that translates source code from one high-level computer language to the source code of another language, but does not explicitly disclose the first computer

language is Java and the second computer language is C#. However, Simser discloses an analogous translating system that translates bi-direction between Java, C, and C++ (Col 2, line 19). It would have obvious to one having an ordinary skill in the art at the time the invention was made to combine Simser's approach and Andrews's approach. Therefore, one having an ordinary skill in the art would have been motivated to translate Java programming language to C# programming language since both of them are high-level computer languages and C# is a cocktail of Java.

Claim 42: Andrews discloses the computer language translating system as in claim 34 above that translates source code from one high-level computer language to the source code of another language, but does not explicitly disclose the first computer language is C# and the second computer language is C++. However, Simser discloses an analogous translating system that translates bi-direction between Java, C, and C++. It would have obvious to one having an ordinary skill in the art at the time the invention was made to combine Simser's approach and Andrews's approach. Therefore, one having an ordinary skill in the art would have been motivated to translate C# programming language to C++ programming language since both of them are high-level computer languages and C# is a cocktail of C++.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Phillip H. Nguyen whose telephone number is (571)

270-1070. The examiner can normally be reached on Monday - Friday 10:00 AM - 3:00 PM EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571) 272-3719. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

PN
10/23/06

Kakali Chaki
Supervisory Patent Examiner

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100